AIAA JOURNAL
Vol. , No. ,

Methodology for Dynamic Data-Driven Online Flight Capability Estimation

Marc Lecerf,* Douglas Allaire,† and Karen Willcox‡
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

DOI: 10.2514/1.J053893

This paper presents a data-driven approach for the online updating of the flight envelope of an unmanned aerial vehicle subjected to structural degradation. The main contribution of the work is a general methodology that leverages both physics-based modeling and data to decompose tasks into two phases: expensive offline simulations to build an efficient characterization of the problem and rapid data-driven classification to support online decision making. In the approach, physics-based models at the wing and vehicle level run offline to generate libraries of information covering a range of damage scenarios. These libraries are queried online to estimate vehicle capability states. The state estimation and associated quantification of uncertainty are achieved by Bayesian classification using sensed strain data. The methodology is demonstrated on a conceptual unmanned aerial vehicle executing a pullup maneuver, in which the vehicle flight envelope is updated dynamically with onboard sensor information. During vehicle operation, the maximum maneuvering load factor is estimated using structural strain sensor measurements combined with physics-based information from precomputed damage scenarios that consider structural weakness. Compared to a baseline case that uses a static as-designed flight envelope, the self-aware vehicle achieves both an increase in probability of executing a successful maneuver and an increase in overall usage of the vehicle capability.

Nomenclature

A	=	upper bound of box constraint for regularized support vector machine
a	=	random variable for quantity a
C	=	capability set
$\hat{\boldsymbol{C}}$	=	probablistic support vector machine output
$\boldsymbol{c}$	=	capability classifier parameter vector
D	=	damage configuration space
D_j	=	event that damage case j occurs
DSR	=	damage library down-sampling ratio
$\boldsymbol{d}$	=	damage parameterization vector
d_f	=	severity of damage on wing
d_t	=	depth of damage on wing
$E[\cdot]$	=	expectation
$\boldsymbol{e}$	=	observable vector measurement noise
f	=	failure metric
K	=	kernel function
l_c	=	chordwise location of damage on wing
l_s	=	spanwise location of damage on wing
N_c	=	number of capability parameters
N_d	=	number of observable vector data samples
N_s	=	number of sensors
n	=	load factor
$n_{\max}$	=	maximum load factor before failure
$n_{\max}^{\text{truth}}$	=	truth reference maximum load factor
n_{op}	=	operational load factor decided upon using dynamic capability estimate
$n_{\text{op}}^{\text{static}}$	=	operational load factor decided upon using static capability estimate from design
$\bar{n}_{\text{util}}$	=	average utilization of maximum vehicle capability
$p(\cdot)$	=	probability
p_{op}	=	threshold probability for decisions using dynamic capability estimate
R	=	number of damage library records
$\boldsymbol{S}$	=	support vector machine discriminant function ("score")
$\mathcal{S}$	=	observable vector space
$\boldsymbol{s}$	=	observable vector
$\hat{\boldsymbol{s}}$	=	observable vector measurement
V	=	airspeed
w_c	=	chordwise extent of damage on wing
w_s	=	spanwise extent of damage on wing
$\mathcal{X}$	=	vehicle state space
$\boldsymbol{x}$	=	vehicle state
$\boldsymbol{\alpha}_j$	=	support vector machine weight for jth training sample
β_1, β_2	=	probabilistic support vector machine regression model parameters
ϵ^k	=	kth strain gage rosette output
ε^k	=	plane strain at location of kth strain gage rosette

I. Introduction

A SELF-AWARE aerospace vehicle can dynamically adapt the way it performs missions by gathering information about itself and its surroundings and responding intelligently. This concept has the potential to improve vehicle performance over the full lifecycle; not only can the system plan and operate independently of human operators, but it can also quantify the state of its available internal resources and maintain knowledge of its current health beyond its initial baseline performance [1]. In this way, the system mimics the behavior of a biological organism; it can act aggressively when it is healthy and in favorable conditions and can become more conservative as it ages and degrades.

There are several challenges associated with enabling a self-aware vehicle. Among them is the task of allowing dynamic updating of the current vehicle structural capability in the case that it undergoes rapid change due to damage or other in-flight events. Methods and tools for dynamic capability estimation have emerged from an intersection of work in both the vehicle damage detection and vehicle design communities. Operational loads monitoring (OLM) aims to improve the detection of damage and fatigue in vehicle structural members. In OLM, onboard aircraft sensors gather structural loading information to identify damage and fatigue (most often postflight) in order to

Presented as Paper 2014-1175 at the 16th AIAA Non-Deterministic Approaches Conference, National Harbor, MD, 13–17 January 2014; received 2 September 2014; revision received 2 April 2015; accepted for publication 14 April 2015; published online XX epubMonth XXXX.

*Graduate Student, Department of Aeronautics and Astronautics. Student Member AIAA.

†Assistant Professor, Department of Mechanical Engineering, currently Texas A&M University, College Station, TX 77845. Member AIAA.

‡Professor, Department of Aeronautics and Astronautics. Associate Fellow AIAA.

reduce maintenance costs and increase reliability [2,3]. At the systems level, the integrated vehicle health management (IVHM) field involves frameworks that incorporate multiple sources of operational data, physics-based models, and prognosis techniques [4,5]. Modern IVHM architectures have been developed at NASA [6] and the Department of Defense [7]. Damage and fault tolerance are now becoming important components of real-time software architectures for monitoring aircraft component health, such as that proposed by the Onboard Active Safety System (ONBASS) project [8]. IVHM has also begun to enter the initial aerospace vehicle design in which unit costs are high, and optimization techniques have been explored to improve IVHM architectures [9].

In addition to systems-level health monitoring, there has been active work at the vehicle component level, particularly in structural composites. Structural health monitoring (SHM) using statistical inference techniques has seen active progress. A broad survey of the SHM field up to 2001 is presented in [10]. Recent work has approached damage detection problems using pattern recognition techniques [11]. Damage identification based on structural vibration data has had particular success [12], in which changes in vibration modal frequencies often denote acute material degradation. More recently, high-fidelity modeling can enter into the damage identification and health management control loop; candidate models of system behavior can be weighted based on real-time data, and actions can be performed to increase estimation confidence as well as to "heal" the system, given current damage estimates [13].

However, work still remains to connect damage parameter identification to the online estimation of quantifiable vehicle capability. There is a need for global metrics used during the design phase (metrics that drive the performance requirements of the vehicle) to be tracked and updated throughout the vehicle's lifetime. Standard design principles for aircraft operate on systems-level analyses such as the V-n diagram [14], in which large margins of safety (often based on empirical evidence and experience) are substantial drivers for design decisions. Current work in condition-aware aircraft maneuverability is developing approaches to replace safety margins with physics-based reasoning [15,16]; however, open questions exist regarding how to integrate local damage identification with updates to global aircraft performance metrics. Forming this connection will improve lifetime usage of assets and could enable designs that rely on dynamic usage patterns in the presence of degradation, i.e., vehicles of which the operational usage changes with their changing physical condition. In this paper, we focus on the setting in which the vehicle undergoes a sudden change in structural properties and there is a need to rapidly update the flight envelope. A specific example is when a unmanned aerial vehicle undergoes a sudden damage event during flight (e.g., under combat circumstances) but needs to complete its mission. Another example is when repair is not an option (e.g., during a long endurance flight). Existing SHM techniques mostly focus on replacing inspections that are associated with scheduled repair or replacement of structural elements that are subject to slow damage growth, and so computational efficiency is not an issue for these methods.

Our goal is to develop a data-driven methodology that receives input from vehicle sensors in flight and rapidly provides updated estimates of vehicle capability with quantified uncertainty. Our approach is based on an offline/online decomposition of tasks that leverages the relative strengths of data and predictive physics-based models. In the offline phase, we create a library of damage cases by running simulations of different damage scenarios and their associated impact on the vehicle flight envelope. In the online phase, we acquire data, use Bayesian classification to estimate in which library case the vehicle might be, and then rapidly update the flight envelope. Our goals and approach differ from classical SHM in two significant ways. First, we do not use sensor information to directly infer damage; rather, we use sensor information to perform a classification by comparing to simulation data from precomputed scenarios. Second, our ultimate prediction goal is not characterization of the damage per se but rather an updated vehicle flight envelope. These differences mean that the demands on sensing technology are different from those in classical SHM; in particular, our results will show that information from strain sensors is sufficiently rich to perform the classification task. Although in this paper we focus on the specific case of sensed structural information leading to a dynamically updated flight envelope, our contribution is a methodological framework that applies in many situations in which dynamic data might inform updated estimates that support improved decision making (e.g., power management based on available fuel or battery levels or engine management based on environmental conditions or fuel composition). In the case considered in this paper, our approach proceeds as follows: Offline, we evaluate loss of structural rigidity due to structural weakness, using physics-based models to construct a library of strain, maneuver, and damage cases. This behavioral library can then be queried online using a Bayesian classification process to determine probable damage and vehicle capability states and to rapidly update the flight envelope.

To demonstrate our methodology, we consider potential structural weakness of the wing of an UAV and quantify how a dynamic data-driven capability estimate could improve vehicle survivability and utilization. A dynamically updated structural capability estimate in flight could improve the likelihood of mission success in contested hostile environments, as the vehicle could immediately replan its mission to account for changes in its structural capability. It could also enable avoidance of maneuvers that would otherwise result in decreased aircraft lifetime and increased maintenance costs. Thus, the potential benefits of a dynamic data-driven capability estimate include mitigation of both mission-related risk and fiscal risk.

The remainder of the paper is organized as follows. Section II presents the general data-driven methodology for capability estimation. Section III presents a representative aircraft model incorporating wing structural weakness; in Sec. IV, we apply our methodology to this model via a classification process. Section V demonstrates how the capability estimate could be used in an online scenario, presenting results with both qualitative and quantitative analyses. Finally, Sec. VI provides concluding remarks.

II. Methodology

Our approach to flight capability estimation relies on a decomposition of computational effort between offline and online phases. The offline phase occurs before operation of the system of interest, when we are able to leverage powerful computational environments that have relaxed execution time and storage constraints. The online phase refers to the real-time (or simply time- and memory-constrained) parts of system operation, when embedded computation needs to be lightweight. We use physics-based models, experimental data, and other sources of information about the system in the offline phase to build approximations of the system behavior; the approximations can then run in the online phase to improve performance, by providing a predictive lens through which to interpret and make use of online sensor data. Sections II.A and II.B describe the offline and online phases of our approach, respectively.

A. Offline Phase

Figure 1 presents a functional decomposition of the offline phase. The process is broken into three stages: characterization of the vehicle using models and/or experiments, classification of vehicle behavior based on failure modes, and storage of these classifiers as records in a behavioral library. The following subsections step through these stages in further detail.

1. *Step One: Characterize System*

Figure 1a shows the first step. The user begins with vehicle system models and/or experiments that represent the vehicle behavior. There are two system inputs and two outputs, the definitions of which are as follows:

1) The state vector $\boldsymbol{x} \in \mathcal{X}$ contains quantities that specify the configuration of the vehicle before considering changes to capability. For a maneuvering aircraft, $\boldsymbol{x}$ could be the kinematic state vector; for instance, in Sec. III, we consider an aircraft in steady flight with a state quantified by an airspeed and a wing load factor, where $\mathcal{X} \subset \mathbb{R}^2$.

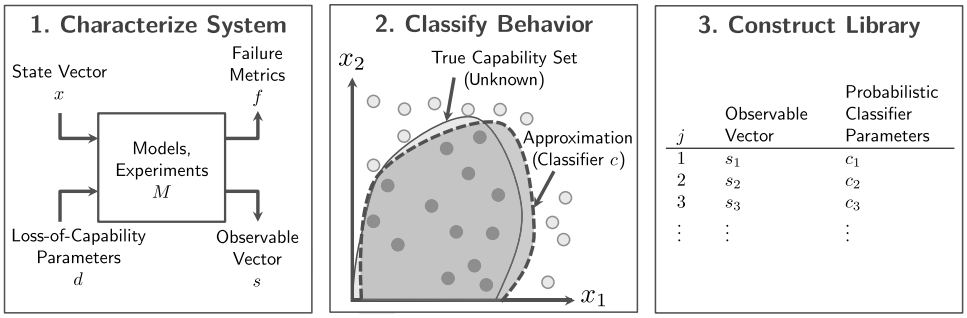


Fig. 1 The three steps in the offline phase for building a library that can be queried in the online phase.

2) The loss-of-capability parameters $\boldsymbol{d} \in \mathcal{D}$ specify how the vehicle could become modified such that its capability set would change. Examples are parameters describing structural damage and parameters describing available system resources such as battery levels or fuel stores.

3) The failure metric f: $\mathcal{X} \times \mathcal{D} \to \mathbb{R}$ provides a measure for how close the vehicle is to undesirable behavior. Examples are closeness of structural loads to maximum thresholds and closeness of available system resources to minimum safe levels.

4) The observable vector $\boldsymbol{s}$: $\mathcal{X} \times \mathcal{D} \to \mathcal{S}$ contains quantities that are available online to provide information about the vehicle state. For example, in Sec. III, we consider an aircraft with N_s continuous strain measurements provided by embedded wing sensors, where $\mathcal{S} \subset \mathbb{R}^{N_s}$.

2. *Step Two: Classify Behavior*

The models and experiments produced in step 1 enable a formulation of the vehicle capability set as follows: If we represent a constraint on the vehicle behavior as an upper bound on the value of the failure metric $f(\boldsymbol{x}, \boldsymbol{d})$ for any input state $\boldsymbol{x}$ and loss-of-capability parameters $\boldsymbol{d}$, then for a fixed value of $\boldsymbol{d}$, the capability is the set of safe states $\boldsymbol{x}$ of which the f values lie below this limit. More precisely, let M_f be the upper bound on f that represents a constraint on the vehicle behavior. Then, the capability set $\mathcal{C}$ is a function of $\boldsymbol{d}$ as follows:

$$\mathcal{C}(\boldsymbol{d}) = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}, \boldsymbol{d}) < M_f\} \tag{1}$$

In general, evaluating $\mathcal{C}$ as a set-valued quantity is intractable, since it requires sampling the entire state space. An approach for making the problem tractable is to use sampling-based classification to approximate $\mathcal{C}$; this technique is represented conceptually in Fig. 1b for a two-dimensional (2-D), continuous state space $\mathcal{X}$ characterized by the state coordinates $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. For each fixed value of $\boldsymbol{d}$, samples are generated from $\mathcal{X}$ and labeled as safe (gray) or unsafe (light gray) based on whether they satisfy or do not satisfy, respectively, the predicate for membership in $\mathcal{C}$ given by expression (1). The labeled samples are then used to train a classifier that designates new query state vectors as safe or unsafe. Because the classifier is trained using a finite set of samples, it can only approximate the true underlying capability set to some finite accuracy, but once the classifier is trained, it could in theory classify every point in the state space; this would produce the approximation to the true capability set as shown (notionally) in Fig. 1b.

The possible discrepancy between the classifier and the true capability set is a form of model error, and we introduce uncertainty to account for it. In particular, we train a probabilistic classifier for each value of $\boldsymbol{d}$ to evaluate the probability that a query state vector belongs to $\mathcal{C}(\boldsymbol{d})$. We denote the quantities needed to implement the probabilistic classification for a given input $\boldsymbol{x}$ as the probabilistic capability classifier quantities $c(\boldsymbol{d})$. In Sec. IV, we implement this form of classification using a probabilistic support vector machine (PSVM), where $c(\boldsymbol{d})$ contains quantities such as support vectors, weights, and distribution hyperparameters.

3. *Step Three: Construct Library*

Step 2 approximates the capability set for each value of the loss-of-capability parameters via sampling-based classification. Now, by combining the samples produced from these runs, we produce a library of records containing the following features: 1) $\boldsymbol{x}$, the value of the system state vector; 2) $\boldsymbol{d}$, values of the system loss-of-capability parameters; 3) f, the value of the system failure metric; 4) $\boldsymbol{s}$, the value of the system observable vector; and 5) $\boldsymbol{c}$, values of the probabilistic capability classifier quantities.

We let R represent the number of records in this library, and we assign subscripts to denote these features for a given record $j = 1, \ldots, R$ as $\boldsymbol{x}_j$, $\boldsymbol{d}_j$, f_j, $\boldsymbol{s}_j$, and $\boldsymbol{c}_j$.

Because the capability set for record j, characterized by $\boldsymbol{c}_j$, depends only on $\boldsymbol{d}_j$, multiple records could have the same values for $\boldsymbol{c}_j$. These records would represent cases in which the system is in the same loss-of-capability case but has a varying system state vector; thus, even though they have the same value of $\boldsymbol{c}_j$, the corresponding observable vector value $\boldsymbol{s}$ will vary.

We store this library for later queries in the online phase, as represented in Fig. 1c. As we will show in the next section, the only features necessary for queries in the online phase are the observable vector $\boldsymbol{s}$ and the probabilistic classifier parameters $\boldsymbol{c}$; the vehicle state and the loss-of-capability parameters are hidden data that are necessary only for modeling the vehicle behavior. Our stored library contains records that provide a direct link from vehicle observable quantities to vehicle capability.

B. Online Phase

In the online phase, we directly infer the vehicle capability from a sensed sample of the vehicle observable vector, by use of the stored vehicle behavioral library. There are two classification steps involved:

1) The observable vector sample (i.e., sensor information) is used to classify the current vehicle behavior into cases represented in the library. We formulate this classification in a Bayesian sense, in which the goal is to minimize the probability of misclassification.

2) Using the probabilistic classifiers that were precomputed and stored for each record in the library, we retrieve the probability that a query vehicle state lies within the current capability set.

The following sections describe the process mathematically. We begin with a description of relevant notation and then formulate the inference process.

1. *Notation and Assumptions*

Since we will be working with probabilistic quantities, our convention is to denote random variables or vectors using serifed letters (e.g., a, b, and s) and to denote values taken by random variables by corresponding unserifed letters (e.g., a, b, and c). We represent the expectation operator as $E[\cdot]$. We represent probability mass and probability density functions as $p(\cdot)$, where the corresponding discrete or continuous case will be clear from context. In the few cases that the random variable dependence is ambiguous, we revert to a subscript notation; for example, $p_{\mathsf{a}}(a)$ and $p(a)$ both represent the probability (or probability density, if a is continuous) that random variable a takes the value a.

We assume quasi-static vehicle behavior, in which for any instant in time the vehicle state takes some value $\boldsymbol{x} \in \mathcal{X}$. By definition (1), the vehicle capability is a set $\mathcal{C} \subset \mathcal{X}$. The models and/or experiments from the offline phase allow us to build a library of information about the vehicle behavior. Here, we refer to each library record as representing a vehicle behavioral case; note this is distinct from the vehicle state. The notation for features of each record in the library follows that from Sec. II.A.3. In the online phase, we use only a subset of the library data. In particular, we use $\boldsymbol{s}_j$, the vehicle observable vector, and $\boldsymbol{c}_j$, a vector describing the probabilistic classifier, for the jth record for $j = 1, \ldots, R$. Each $\boldsymbol{c}_j$ allows us to compute the probability that a query state $\boldsymbol{x}'$ lies in the capability set corresponding to the jth behavioral case. We write this probability as $p(\boldsymbol{x}' \in \mathcal{C}|D_j)$, where D_j is an indicator event designating whether the vehicle exists currently in the behavioral case represented by the jth library record. The D_j are mutually exclusive; i.e., the vehicle can be in at most one behavioral case at any point in time ($D_j = 1$). However, this does not mean the vehicle is guaranteed to be in any of the library behavioral cases.

Sensors provide measurements of the values in the observable vector $\boldsymbol{s}$. We denote the random vector corresponding to these measurements as $\hat{\boldsymbol{s}}$. Given that the vehicle is in the jth behavioral case, $\hat{\boldsymbol{s}}$ has the form

$$\hat{\boldsymbol{s}} = \boldsymbol{s}_j + \mathsf{e} \tag{2}$$

where e is a random vector representing measurement noise that is independent of the vehicle behavioral case. We assume the user has knowledge of the statistics of e (often for physical systems, it is characterized using a multivariate Gaussian with known mean and covariance); that is, we can compute $p_{\mathsf{e}}(e)$, which leads to

$$p(\hat{\boldsymbol{s}}|D_j) = p_{\mathsf{e}}(\hat{\boldsymbol{s}} - \boldsymbol{s}_j) \tag{3}$$

2. Capability Estimator Formulation

The goal of our inference process is to evaluate the vehicle capability given a measurement of the observable vector $\hat{\boldsymbol{s}}$. Because the vehicle capability is a set, one means of performing this task is to evaluate set membership (as introduced in Sec. II.A.2). That is, we desire to evaluate a function $q\colon \mathcal{X} \times \mathcal{S} \to \mathbb{R}$ that closely approximates the probability of a query state $x' \in \mathcal{X}$ lying within $\mathcal{C}$, given we observe $\hat{\boldsymbol{s}} = \hat{s}$. Mathematically, this is written as

$$q(\boldsymbol{x}', \hat{\boldsymbol{s}}) \approx p(\boldsymbol{x}' \in \mathcal{C}|\hat{\boldsymbol{s}}) \tag{4}$$

In the following, we employ an estimator that combines information from each behavioral case, in which more likely cases have more influence on the overall estimate than unlikely ones. Equation (4) can be expressed as a summation using the Law of Total Probability:

$$p(\boldsymbol{x}' \in \mathcal{C}|\hat{\boldsymbol{s}}) \approx \sum_{j=1}^{R} p(D_j|\hat{\boldsymbol{s}})\, p(\boldsymbol{x}' \in \mathcal{C}|D_j, \hat{\boldsymbol{s}}) \tag{5}$$

The expression (5) is an approximation because it relies on an assumption that $\sum_{j=1}^{R} p(D_j|\hat{\boldsymbol{s}}) = 1$, i.e., that our current vehicle behavioral case is contained somewhere in the R records in our library. This is an approximation that becomes increasingly accurate as our library becomes larger and richer.

When conditioned on D_j, $\{\boldsymbol{x}' \in \mathcal{C}\}$ is independent of $\{\hat{\boldsymbol{s}} = \hat{s}\}$ because the sensor noise is assumed to be independent of the vehicle behavioral case [see Eq. (2)]. Thus, we can drop the conditioning on $\hat{\boldsymbol{s}}$ in the second term inside the summation on the right-hand side of Eq. (5). Then applying Bayes's rule, we obtain a final expression for our capability estimator $q(\boldsymbol{x}', \hat{\boldsymbol{s}})$:

$$q(\boldsymbol{x}', \hat{\boldsymbol{s}}) = \frac{\sum_{j=1}^{R} p(\hat{\boldsymbol{s}}|D_j)\, p(D_j)\, p(\boldsymbol{x}' \in \mathcal{C}|D_j)}{\sum_{j'=1}^{R} p(\hat{\boldsymbol{s}}|D_{j'})\, p(D_{j'})} \tag{6}$$

In Eq. (6), the term $p(\boldsymbol{x}' \in \mathcal{C}|D_j)$ is the value of the probabilistic classifier for behavioral case j for the query vehicle state $\boldsymbol{x}'$. The product $p(\hat{\boldsymbol{s}}|D_j)p(D_j)$ can be interpreted as a weighting term, while the denominator of the right-hand side provides a normalizing factor. Thus, q can be interpreted as a weighted sum of the predictions that would be made by each record individually in the library were we to assume the vehicle was in each record's behavioral case. Probability distributions of this form are called mixture distributions, in which they are derived as a weighted summation of the distributions of distinct, underlying random variables. These underlying variables are often called mixture components, and their weights are often called the mixture weights.

The weighting term $p(\hat{\boldsymbol{s}}|D_j)p(D_j)$ in Eq. (6) requires knowledge of $p(D_j)$, the prior probability that the vehicle is in the jth behavioral case. In the example in Sec. V, we set $p(D_j)$ as a maximum-entropy, uniform prior over all j; however, the user could choose to use a different distribution to encode domain-specific prior knowledge about the vehicle behavior.

3. Scalability

The online phase needs to be cognizant of available computational resources; we analyze the complexity of the methodology and discuss the implication with respect to its practical usability. The computational runtime complexity of the Bayesian classifier can vary significantly depending on the application. Duda et al. [17] present a detailed analysis for the case in which the noise model is a multivariate Gaussian; we present an abbreviated form here. In our case, the jth record of the lookup table represents a distinct class in which the output noise model for said class is $p(\cdot\,|\boldsymbol{s}_j) \sim \mathcal{N}(\boldsymbol{s}_j, \Sigma)$ for some known covariance matrix Σ. The computational runtime complexity follows from Eq. (6):

1) Computing $p(\hat{\boldsymbol{s}}|D_j)p(D_j)$, for the multivariate Gaussian case, the probability of seeing output $\hat{s}$ from class D_j takes the following form:

$$p(\hat{\boldsymbol{s}}|D_j) = \frac{1}{\sqrt{(2\pi)^{N_s}|\Sigma|}} \exp\left[-\frac{1}{2}(\hat{\boldsymbol{s}} - \boldsymbol{s}_j)^T \Sigma^{-1} (\hat{\boldsymbol{s}} - \boldsymbol{s}_j)\right] \tag{7}$$

Given that each observable vector has N_s elements, computation of $\hat{\boldsymbol{s}} - \boldsymbol{s}_j$ is $\mathcal{O}(N_s)$ and multiplication by Σ^{-1} is $\mathcal{O}(N_s^2)$ (computation of Σ^{-1} only needs to be performed once and does not grow with N_s). Overall, the complexity of this step is $\mathcal{O}(N_s^2)$.

2) Computing $p(\boldsymbol{x}' \in \mathcal{C}|D_j)$, we must evaluate the capability boundary for each lookup table record that has a nonzero probability given the sensor data. This will grow at most linearly with the number of records R because the computation for each record depends only on the information within that record. If we denote the complexity of evaluating the capability boundary for a single record as O_c, then the complexity of this step can be expressed as RO_c.

In summary, the computational runtime complexity of the estimator grows as $\mathcal{O}(RN_s^2 O_c)$, where O_c is the complexity of performing a single capability boundary evaluation (which is problem dependent). Adding sensor measurements (i.e., increasing N_s) has a greater impact on the runtime than increasing the number of records in the library; however, the methodology allows for an arbitrarily large library, making R an important component of the runtime complexity growth. The storage requirements for the estimator grow linearly with the initial size of the library, i.e., as $\mathcal{O}(R(N_s + N_c))$, where N_c is the number of elements in the capability parameter vector $\boldsymbol{c}$.

III. Aircraft Capability Model

The methodology developed in the previous section can be applied in a number of different settings. We now tailor it to the specific case of aircraft capability estimation. Before presenting the details of the methodological approach for this setting, we first present the physics-based models that are used in the offline phase. Section III.A presents a conceptual UAV design that is used as a case study. Section III.B describes a vehicle-level model for estimating loads, and Sec. III.C presents a model for representing local wing structural weakness. Section III.D describes the overall coupled vehicle model and defines

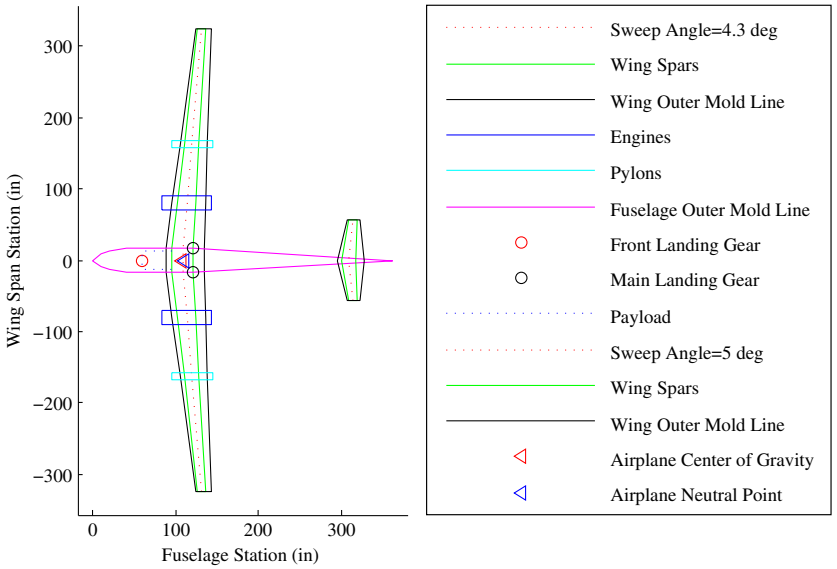


Fig. 2 A realistic concept unmanned aerial vehicle established in order to estimate the effect of structural change on capability.

the specific inputs and outputs, the state, damage, observable, and failure parameters, for the aircraft capability estimation setting.

A. Aircraft Design

Figure 2 shows a conceptual UAV design established using a first-principles sizing routine [18] and Federal Aviation Regulation (FAR) 23[§] guidelines. As shown in the figure, the vehicle has a wing span of 55 ft. It is estimated to cruise at 140 kt (240 ft/s) at an altitude of 25,000 ft. A payload of 500 lb is allowed for in the fuselage. The range of the aircraft is estimated to be approximately 2500 n mile, corresponding to a duration of 17.5 h and allowing for adequate operational capability to explore maneuverability as a function of the changing structural state of the vehicle.

B. Global Aircraft Kinematics

Loads on the UAV are estimated using an ASWING model of the aircraft. ASWING is a nonlinear aerostructural solver for flexible-body aircraft configurations of high to moderate aspect ratio [19]. We use it here to predict internal wing stresses and deflections as a function of input aircraft kinematic states and estimates of changes to the nominal aircraft structure. Figure 3 shows the representation of our concept UAV in the ASWING framework. The ASWING model is a set of interconnected slender beams, one each for the wing, fuselage, horizontal stabilizer, and vertical stabilizer. Lifting surfaces (the wing and stabilizers) have additional cross-sectional lifting properties that are prespecified. We are able to obtain internal aircraft structural loads for static and dynamic flight conditions; however, we restrict ourselves here to analyzing quasi-static pullup maneuvers.

We demonstrate our methodology by representing structural degradation due to damage on the aircraft wing. Damage is considered in a limited sense here only as a structural weakness, represented as a reduction in material stiffness properties. This is a simplified approximation to a full damage model and captures the loss of ability to carry structural loads in the damaged region. We are concerned with the effect a damage situation would have on the vehicle behavior and capability and on the resulting redistribution of loading within the wing; we do not capture the exact shape and nature of the damage event itself. However, we note that our general approach could extend to handling more complex damage models, at the cost of additional computation.

[§]United States Code of Federal Regulations, Title 14, Part 25, Sect. 25.105–25.119.

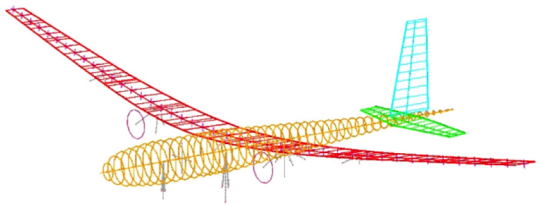


Fig. 3 The ASWING representation of our concept UAV [19]. The structure is specified as a set of interconnected slender beams, where lifting surfaces have additional aerodynamic properties specified along their span.

C. Local Wing Structural Weakness Representation

To resolve stiffness loss due to local structural weakness on the aircraft wing, we need another technique to interface with the global ASWING aircraft model. In lieu of forming a computationally expensive, full three-dimensional (3-D) finite-element representation of the wing, we use variational asymptotic beam cross-sectional analysis (VABS) [20], a powerful dimension reduction technique used in industry practice. A visual representation of the VABS technique is shown in Fig. 4. The beam of interest is modeled via an array of two-dimensional cross-sectional finite-element models (FEMs). The cross-sections can capture the details of a multiply composite wing skin and local structural weakness effects. VABS computes lumped stiffness and inertial properties at a reference point in each cross-section, forming a global line representation of the beam. A standard beam problem solver is used to find the global force and moment distribution along this reference line given input forces and moments. Finally, using the reference line solution, the internal strain field is recovered in the beam cross-sections using relations initially computed by VABS.

In this work, we use the specific UM/VABS implementation developed in FORTRAN by R. Palacios and C. Cesnik at the University of Michigan [21]. Our beam of interest is the aircraft wing box, and ASWING manages the one-dimensional beam solution, computing loads in the wing box for specific flight conditions.

We assume a constant cross-section for our wing of interest and view damage events on the wing surface as quasi-rectangular, constant-depth regions. At the location of the damage, additional cross-sections are added to the beam description to capture the modified wing box properties. An example two-dimensional wing box cross-section model input to VABS is shown in Fig. 5. The wing box follows the shape of a DA-01 airfoil with a chord of 50 in. The ribs are located at 20% and 70% of the chord length (with respect to the leading edge of the airfoil). The wing box has a stack of five plies

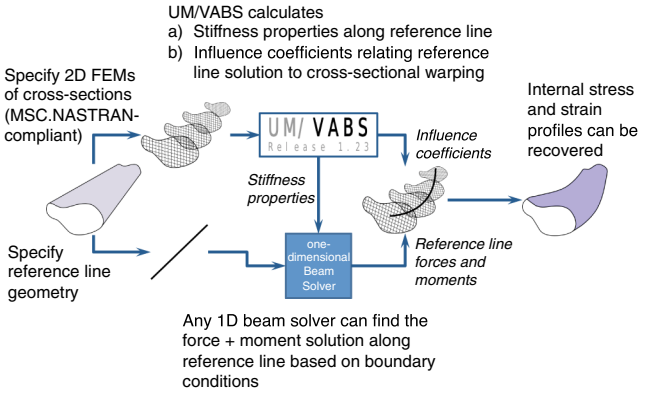


Fig. 4 VABS allows for dimensional reduction of an expensive three-dimensional beam solution into 2-D finite-element models coupled with an external beam solver [20].

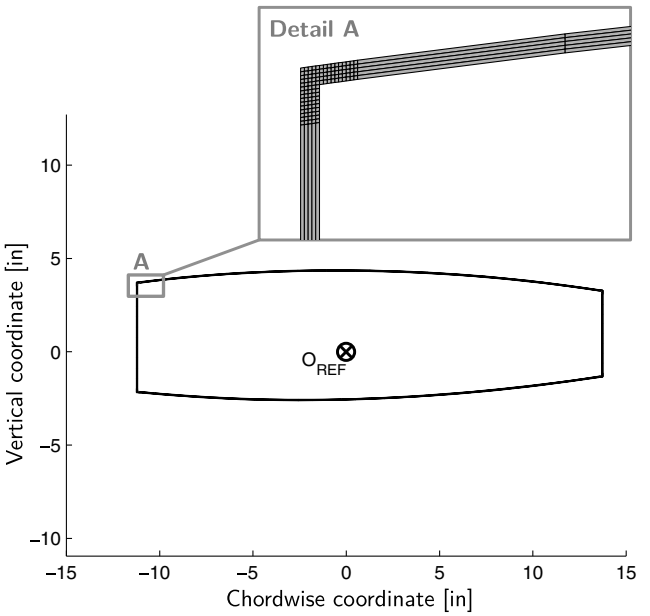


Fig. 5 Example 2-D finite element model of a spanwise wing box cross-section. Detail A shows the ply stack. VABS computes lumped stiffness and inertial properties at the reference point O_{REF}.

made of AS4/MTM45-1 material, with orientations $[0/+45/90+45/0]_T$ deg (where the subscript T denotes total laminate).

To model the behavior of structural sensors, we extract a subset of the three-dimensional wing box strain solution for given maneuver and damage conditions. For our application, we consider strain gage rosettes, which are capable of reading in-plane surface strains at point locations on the wing box. We will elaborate on the strain gage model further in Sec. V; for now, it suffices to say that the model outputs plane strain values at select locations on the wing box surface as shown in Fig. 6. The figure shows that we make the assumption that the strain sensors are located in close proximity to the region in which structural weakness is introduced (orange shaded region). This permits us to validate the capability estimation process in the case that sensor measurements show noticeable changes due to the introduced structural weakness. This is important since disturbances in the strain field will be local to the damaged area. Again, we emphasize that our methodology relies on recognition of changed vehicle structural capability, not on detection of the damage itself; thus, local strain sensors can provide useful information to inform the classification process even if they cannot detect the type and specific location of the damage itself. Nonetheless, the optimal choice of sensor technology and placement is an important question for future work, as is the question of how to identify and process sensor measurements that may themselves be affected by damage.

We identify unsafe structural behavior as when part of the aircraft wing experiences strains that exceed maximum strains for known failure modes; in our case, these are extensional and shear strain limits with respect to composite material axes. We quantify this with the failure index, defined here as the ratio of the current strain in a structural element to the strain allowable for the element material. A single element will have multiple failure indices, each corresponding to a specific component of the element's strain tensor and its maximum allowable value. We consider only elements that do not lie specifically in our damage region (i.e., we do not consider those elements of which the stiffness properties have been modified) because we are not modeling the behavior of the damage itself, and the strains computed by our model inside this region may not be of physical significance. To derive a representative failure metric that encompasses the behavior of the entire wing, we 1) extract the failure index corresponding to each failure mode for each element of the wing, not considering those elements of which the stiffness properties were artificially modified, and 2) find the maximum failure index over all modes and all elements.

We denote the final scalar result as f. It is an upper bound on all other failure indices in the wing. The case in which $f \geq 1$ corresponds to an unsafe situation in which the allowable strain has been exceeded somewhere in the wing, and the case in which $f < 1$ corresponds to a safe situation in which all elements of the wing are experiencing strain that is below their allowable thresholds.

D. Coupled Vehicle Model

To construct the offline library, we input maneuver and damage conditions into a vehicle model consisting of the coupled ASWING and VABS models as shown in Fig. 7. Each of the inputs and outputs is described as follows:

1) The state vector $\boldsymbol{x} = (V, n)$ input to the vehicle model specifies a quasi-static pullup maneuver at an airspeed V and load factor n (i.e., the ratio of lift to weight).

2) The damage parameters input into the vehicle model represent a quasi-rectangular region on the aircraft with some fixed depth in

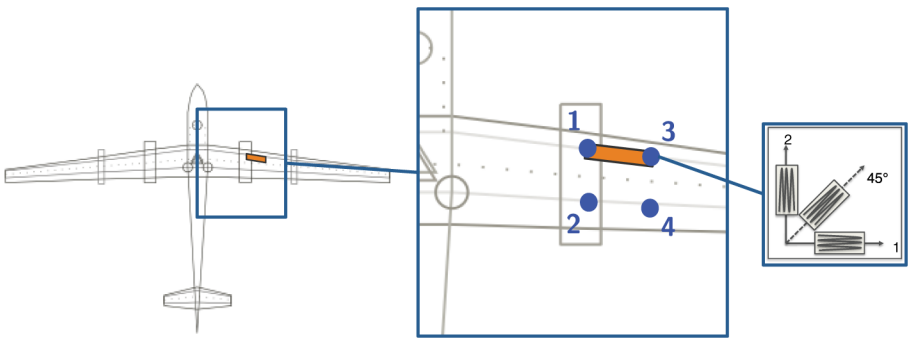


Fig. 6 The locations of the strain gages on the wing box top surface in the aircraft model. The inset on the right shows the "rectangular" variant of strain gage rosette.

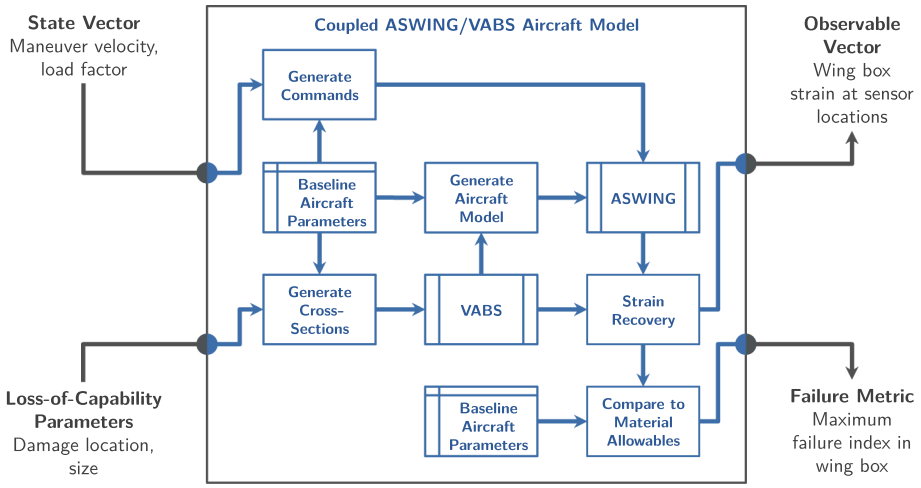


Fig. 7 Flow of information through aircraft model. VABS handles the local damage to the wing structure, while ASWING computes the global aircraft structural solution as a function of input maneuver conditions.

which material stiffnesses are decreased uniformly. Specifically, damage is introduced as structural weakness and is characterized by the vector $\boldsymbol{d} = [l_s, w_s, l_c, w_c, d_t, d_f]^\top$, illustrated in Fig. 8. The elements l_s and w_s are, respectively, the spanwise location and width with respect to body axes; l_c and w_c are, respectively, the chordwise location and width on the aircraft wing with respect to the leading edge at l_s; and d_t is the depth of the region of structural weakness with respect to the local normal of the top surface of the wing. The element $d_f \in [0, 1]$ is a fraction representing the severity of the structural weakness, for which the material stiffness values are reduced relatively by d_f in the damaged area.

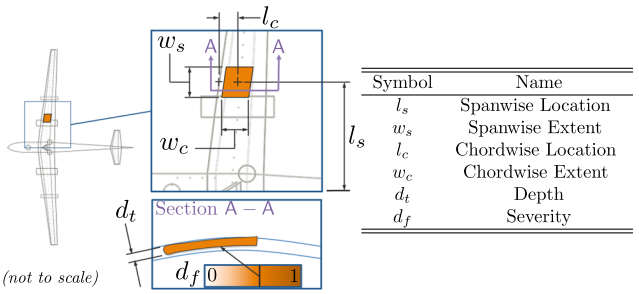
3) The observable vector $\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{d})$ output from the model is a concatenation of plane strain values from select sensor locations on the wing box top surface.

4) The failure metric $f(\boldsymbol{x}, \boldsymbol{d})$ output from the model is a maximum failure index in the wing structure as described previously in Sec. III.C.

As shown in Fig. 7, the damage parameters are used to interface with VABS to compute wing cross-sectional structural properties for use in the ASWING model. The state vector, which describes the global aircraft maneuver, is input to the ASWING model to analyze the maneuver scenario for the modified vehicle model. The ASWING estimated loads are then combined with influence relations pre-computed by VABS to obtain the three-dimensional internal wing strain field. From this strain field, we compute the failure indices over undamaged elements (i.e., over elements outside of the region shown in Fig. 8) and the values of the observable vector quantities.

IV. Characterizing Capability via Classification

This section tailors the general methodology proposed in Sec. II to the case of characterizing vehicle capability for the UAV model presented in Sec. III. Section IV.A describes our classification approach based on probabilistic support vector machines, and Sec. IV.B presents an adaptive state-space sampling approach.



Symbol	Name
l_s	Spanwise Location
w_s	Spanwise Extent
l_c	Chordwise Location
w_c	Chordwise Extent
d_t	Depth
d_f	Severity

Fig. 8 Illustration of the parameters characterizing structural weakness in our aircraft model. We bound damage events using a quasi-rectangular region on the aircraft surface with some fixed depth.

A. Approximation Using Probabilistic Support Vector Machines

Given a damage case applied to our aircraft model, we sample in the maneuver state space, labeling each sample as safe or unsafe according to the value of its output maximum failure index. As we perform this task, we save the observable vector corresponding to each sample for use later in the online phase. We first describe the construction of a support vector machine (SVM) model and then describe the extension to a probabilistic SVM model that quantifies uncertainty in the classification estimates.

1. *Support Vector Machine*

We use the samples to build a classifier that can approximate the true safe/unsafe label of any point in the maneuver space. This classifier then becomes our representation of the capability of the aircraft given the damage case. In this work, we have chosen to use an SVM-based classification approach, a technique from the machine learning community, because we are motivated by rapid decision making of the yes/no (i.e., safe/unsafe) form. Other choices for the classification are possible, including methods that characterize closeness to a boundary; note that our overall methodological framework is general and not tied to the specific choice of SVMs as the classifier. For example, if we wanted to estimate closeness to the boundary, one choice would be the use of signed distance functions for binary classification [22]. This would provide more information but would be more expensive to employ online. An abbreviated explanation of the SVM technique is presented here; for further detail, we refer the reader to related material by Duda et al. [17] or the original article by Cortes and Vapnik [23].

The SVM performs binary classification of unlabeled test samples (i.e., classification into one of two classes) based on trends seen in a labeled set of training samples. More formally, let our collection of N labeled training samples take the form $\mathcal{Z} = \{(\boldsymbol{x}_j, y_j) : \boldsymbol{x}_j \in \mathcal{X}, y_j \in \{-1, 1\}, j = 1, \ldots, N\}$, where each $\boldsymbol{x}_j$ is a state vector consisting of an airspeed V and load factor n. In general, each $\boldsymbol{x}_j$ consists of attributes that describe the sample. Each y_j is the corresponding binary label for the sample, which in our case is the indicator representing whether the failure metric $f(\boldsymbol{x}_j, \boldsymbol{d})$ (given the fixed damage parameters $\boldsymbol{d}$) exceeds a nominal safe threshold value. We assign a value of $y_j = 1$ if sample j is labeled as safe and a value of $y_j = -1$ if sample j is labeled as unsafe.

The SVM constructs and evaluates a discriminant, S: $\mathbb{R}^n \to \mathbb{R}$, such that the input sample $\boldsymbol{x}$ is labeled -1 if $S(\boldsymbol{x}) \leq 0$ and 1 if $S(\boldsymbol{x}) > 0$. The value of this discriminant for a sample $\boldsymbol{x}$ is often called its score. A simple SVM discriminant is the linear case $S(\boldsymbol{x}) =$

$\boldsymbol{w}^T\boldsymbol{x} + b$, a hyperplane with normal vector $\boldsymbol{w}$ and offset b. We use the nonlinear extension, which can be shown to have the form

$$S(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}) + b \tag{8}$$

where K is a suitable kernel function (this is often called the kernel trick, popularized initially in the machine learning community by Aizerman et al. [24]; in many practical applications, it is a key enabler of the construction of nonlinear SVMs), b is the bias as in the linear case, and the α_j are weights that are nonzero for only a subset of the training samples that lie closest to the decision boundary, called the support vectors. The weights α_j and bias b can be obtained from the dual of the SVM optimization problem:

$$\max_{\alpha}\left[-\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_j\alpha_k y_j y_k K(\boldsymbol{x}_j, \boldsymbol{x}_k) + \sum_{j=1}^{N}\alpha_j\right]$$
$$\text{subjectto } 0 \le \alpha_j \le A \quad \forall\ j$$
$$\sum_{j=1}^{N}\alpha_j y_j = 0 \tag{9}$$

Here, A is an upper bound on what is often called the box constraint that confines the allowable values of the weights α_j. A is a parameter tuned by the user to allow slack in the SVM training process, so some training samples might be misclassified, but the SVM model will be less prone to overfitting.

2. *Probabilistic Support Vector Machine*

There is inherent uncertainty in the classification process given a large but finite set of offline samples. Therefore, we include a means of extending the SVM technique to represent uncertainty in its output, called the probabilistic support vector machine. Here, the trained SVM is postprocessed and fitted with a suitable probability distribution. We follow the original technique as proposed by Platt [25], including modifications proposed by Lin et al. [26] to handle numerical instabilities in Platt's original paper. Several other implementations exist of the PSVM training process, including a modification proposed by Basudhar [27].

Given the data set $\mathcal{Z}$ as described previously, and a support vector discriminant $S(\boldsymbol{x})$ characterizing the decision boundary between the two classes, we consider a probabilistic classifier $\hat{C}\colon \mathbb{R}^n \to \mathbb{R}$ that evaluates $p(y = 1|S(\boldsymbol{x}))$, the probability that the sample $\boldsymbol{x}$ lies in class $y = 1$ given the output of our SVM discriminant.

Platt [25] fits $\hat{C}$ with a sigmoid

$$\hat{C}(\boldsymbol{x}) = p[y = 1|S(\boldsymbol{x})] = \frac{1}{1 + e^{\beta_1 S(\boldsymbol{x}) + \beta_2}} \tag{10}$$

where $\beta_1 < 0$ and β_2 are suitable distribution parameters. Given the restriction on β_1, $\hat{C}$ is monotonic in S, ranging from 0 when $S \to -\infty$ to 1 when $S \to \infty$. This reflects the fact that $\hat{C}$ ought to become confident (i.e., a certain 0 or 1) far from the SVM decision boundary at $S(\boldsymbol{x}) = 0$. We find the values of β_1 and β_2 that maximize the log-likelihood,

$$\sum_{j=1}^{N} t_j\ \log(p_j) + (1 - t_j)\log(1 - p_j) \tag{11}$$

where $p_j = \frac{1}{1+e^{\beta_1 S(\boldsymbol{x}_j)+\beta_2}}$ is the probability that sample j belongs to class $y = 1$ given a particular parameterization (β_1, β_2), and we assume that each training sample is independent and identically distributed. The t_j are defined as

$$t_j = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_j = 1 \\ \frac{1}{N_- + 2} & \text{if } y_j = -1 \end{cases} \tag{12}$$

where N_+ and N_- are the number of samples in class $y = 1$ and the number of samples in class $y = -1$, respectively. As described by Platt, the t_j correspond to a maximum a posteriori estimate of the target probability for each class assuming a uniform, uninformative prior over the probability of all our training samples having the correct label [25].

B. Adaptive State-Space Sampling Technique

Our method requires intelligent sampling of the vehicle state space so that we can provide our PSVM training process with a rich set of training samples while minimizing uninformative calls to our aircraft capability model. We have two competing qualitative goals during the sampling process:

1) We want to sample along the boundary of the vehicle capability set to provide an accurate description of its limits.

2) We want to sample within the vehicle capability set in order to capture the vehicle behavior we expect to see during operation (so that our online classification process sees library records that are similar to the observed vehicle behavior).

Techniques exist to provide a space-filling set of samples for goal 2, such as Latin hypercube sampling [28] or a centroidal Voronoi tessellation (CVT) [29]. Refinement of the boundary itself for goal 1 can be implemented using adaptive sampling; we use a technique developed in Ref. [27]. The algorithm begins with a well-spaced set of samples (here, we start with a CVT produced using Lloyd's algorithm [30]) and then chooses samples at each iteration that lie along the boundary of an SVM approximation of the true capability set. A summary of the algorithm steps is as follows:

1) Begin with an initial set of training samples that has at least one member from each of the two classes.

2) Train a SVM on the initial set of samples.

3) Generate a sample on the SVM boundary that lies as far as possible from all current training samples.

4) Generate a second sample nearby the SVM boundary to prevent SVM locking (see [31] for further explanation).

5) Retrain the SVM using the two new samples.

6) Repeat from step 3 until converged.

As a choice for a convergence criterion, Basudhar and Missoum [32] suggest using a polynomial kernel to construct the SVM for each iteration and looking for a stabilization of the change in polynomial coefficient values between iterations. We use a different criterion that makes use of the computation of the sample from step 3, in which we are maximizing the minimum distance from the new sample to any other training sample, while constraining the new sample to lie along the SVM boundary. This distance itself can be used as a convergence metric, and we terminate the algorithm when it decreases below a nominal value (scaled with respect to the bounds of the whole sample space). The intuition behind this metric is that, as the sampling converges onto a SVM boundary in the sample space, new samples will begin to crowd along the SVM boundary line until the distance metric settles to a small value. This convergence criterion works successfully for this problem and is simple to implement; however, the reader is cautioned that it may in general encounter problems if the SVM changes shape rapidly between iterations so that newly added samples take some time to begin settling to nearby locations. A different solution to address this issue for the Gaussian radial basis function kernel proposed by Basudhar and Missoum uses fixed test points that are set during an initial sampling of the design space, in which the fraction of these points that switch SVM classification label between iterations provides a measure of the SVM convergence.

Using this sampling technique, we determine a SVM representation of the vehicle capability boundary to within a desired level of sampling accuracy and then fit a PSVM model to capture the uncertainty in the boundary location due to the finite sampling accuracy. Figure 9 shows the evolution of the computed PSVM for a single vehicle damage case and increasing numbers of state-space samples. The first plot includes only samples from the initial CVT, whereas the next two plots include samples generated using the adaptive sampling technique. Figure 10 shows the convergence plot for the adaptive sampling technique when applied to the case shown in Fig. 9. The

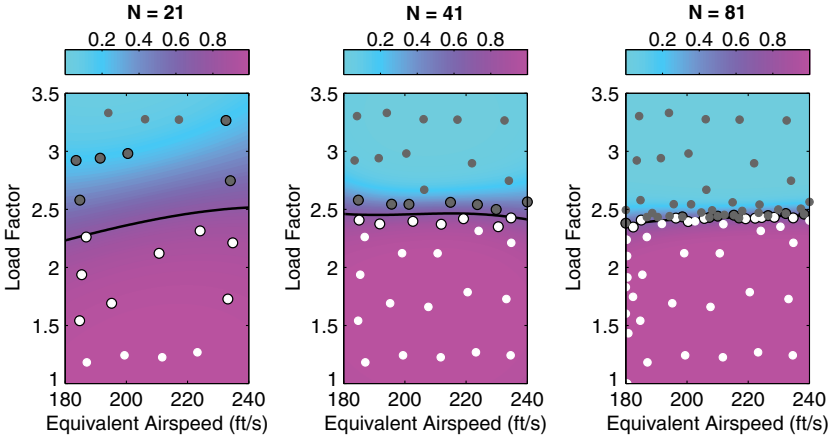


Fig. 9 PSVM vehicle capability estimate for one damage case. White samples classified as "safe," grey samples are "unsafe," black outlines denote support vectors, black line denotes the SVM boundary.

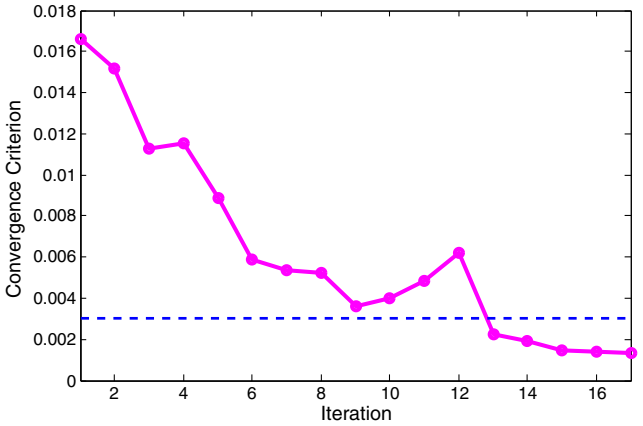


Fig. 10 Convergence history for Basudhar and Missoum's adaptive SVM-based sampling algorithm. The dashed line represents convergence tolerance.

solid line is the value of the convergence metric, and the dashed line is the fixed tolerance value used for the stopping criterion. The resulting samples then yield the data needed to populate the library of records, as described in Sec. II.

C. Using PSVM Library for Online Capability Estimation

As mentioned previously, for each damage case, we form a corresponding PSVM approximation to its capability set. As seen in Eq. (10), each PSVM is parameterized by its values for β_1, β_2, and the SVM support vectors, weights, and bias that define the discriminant $S(\boldsymbol{x})$. In the context of the general methodology from Sec. II, these are the elements that characterize the probabilistic capability classifier parameters $\boldsymbol{c}_j$ for damage case j.

To place the usage of the PSVM within the context of the online estimation process from Sec. II, we revisit the steps to compute the capability estimator $q(\boldsymbol{x}', \hat{\boldsymbol{s}})$ via Eq. (6):

1) Given input sensor data $\hat{\boldsymbol{s}}$, the probability of the vehicle being in each case in the library is computed as $\frac{p(\hat{\boldsymbol{s}}|D_j)p(D_j)}{\sum_{j'=1}^{R} p(\hat{\boldsymbol{s}}|D_{j'})p(D_{j'})}$ for each record j, an application of Bayes's rule.

2) Given a nonzero probability that the vehicle is in the case represented by record j, the probability that the queried state $\boldsymbol{x}'$ is safe $[p(\boldsymbol{x}' \in \mathcal{C}|D_j)]$ is computed. This is given by the output of the PSVM for the jth record in the library, computed via Eq. (10) evaluated at $\boldsymbol{x}'$.

3) The predictions from step 2 are weighted by the probabilities computed in step 1 and summed to produce the final output $q(\boldsymbol{x}', \hat{\boldsymbol{s}})$, as in Eq. (6).

The results in the next section demonstrate the entire flow of this process from sensor data to capability estimates in the simple case of strain measurement data indicating structural weakness. We note, however, that our approach also has utility in the case of known location and type of damage (e.g., in the case that the vehicle has a more sophisticated damage detection system onboard). In this case, even if the damage state is known precisely, the challenge remains to translate this knowledge into a rapid estimate of the current vehicle capability (i.e., to estimate the updated flight envelope). Dynamically analyzing the known damage state with the structural and vehicle simulation models described in Sec. III is impractical, since even this single analysis would be too expensive to achieve in near real time. Instead, our approach uses the precomputed cases considered in the offline phase. Following the three steps outlined previously, the known damage state is matched probabilistically to those damage states considered in the offline training phase. The corresponding PSVMs then provide the rapid mapping to update the vehicle capabilities, albeit with limitations on the accuracy due to the closeness of the offline damage cases to the actual damage case and due to the error in the PSVM approximation of the capability set.

V. Demonstration and Results

This section applies our approach to an example scenario in which the aircraft must perform an evasive pullup maneuver at constant airspeed. Our approach yields data-driven estimates of maximum achievable load factor given structural failure index constraints in the presence of structural weakness. Section V.A describes the problem setup and damage test cases considered. Section V.B discusses behavior of the aircraft capability estimator, while Sec. V.C compares maneuver decision outcomes using our dynamic data-driven estimator to those using a typical static capability estimate. Section V.D analyzes the tradeoffs and uncertainties associated with the dynamic capability estimation.

A. Application Problem Setup

1. *Flight Scenario*

One potential application of our method is for missions in contested environments, in which threats to the vehicle due to hostile agents require a fast, defensive reaction to avoid dangerous regions of the flight zone. In addition, the vehicle may sustain damage on the wing surface that impedes its ability to operate at its initial design capability. Figure 11 presents a schematic of this scenario, in which the vehicle initiates the evasive action at an airspeed of 210 ft/s and an initial load factor of 1.3 (representative of a nominal maneuvering speed and an upper bound on the nominal maneuvering load factor during normal operation while navigating a sequence of waypoints).

2. *Damage Test Cases*

During the offline phase, we construct a library of damage cases and build PSVMs to represent the modified vehicle capability. We generate 150 damage cases (in addition to the nominal case) by performing two full-factorial explorations of the damage parameters at the levels shown in Table 1. We evaluate each combination of

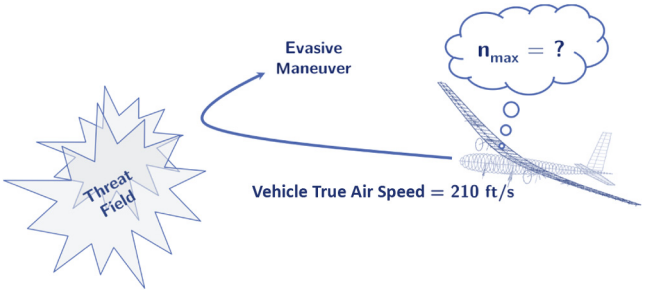


Fig. 11 Schematic of the flight scenario to which we apply our capability estimation framework.

damage parameters for the scenario maneuver (airspeed $V = 210$ ft/s and load factor $n = 1.3$). Note that, since the system is analyzed at a constant vehicle state $\boldsymbol{x}$, the library records have distinct values of $\boldsymbol{d}$, and hence they each have distinct values for $\boldsymbol{c}$. The general methodology presented here can incorporate varying the value of $\boldsymbol{x}$. Note also that grid-based sampling is used for selecting the damage cases included in the library, while for each damage case, adaptive sampling is used in the maneuver space to form the corresponding PSVM.

For assessing our methodology, we focus on five representative damage cases, three from the library and two that are not sampled in the offline phase. The cases are labeled $\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3$, and $\boldsymbol{C}_4$ and are described in Table 2. $\boldsymbol{C}_0$ is the undamaged configuration, while the other four cases represent varying degrees of structural weakness severity. For each of these damage cases, we compute the true maximum allowable load factor $n_{\max}^{\text{truth}}$ at the constant maneuver airspeed 210 ft/s using a simple bisection routine over the load factor n, discovering where the structural failure index f transitions from safe to unsafe. This serves as a baseline to assess the performance of our capability estimate in each damage case.¶

3. *Structural Strain Sensor Model*

To demonstrate and evaluate our methodology, we generate synthetic data by producing model-based estimates of the observable vector for a damage case of interest and corrupting them with noise. In this subsection, we describe a structural strain sensor model, constructed so that this process generates data that are representative of what might be measured in flight.

We model the behavior of strain gage rosettes mounted on the surface of the aircraft wing box to obtain plane-strain measurements, with four measurement locations as shown in Fig. 6. Many varieties of strain gage rosettes exist [33]; here, we consider a rectangular configuration in which two gages placed on the principal composite material axes obtain extensional strains directly and a third placed off axis at 45 deg is used to compute indirectly the in-plane shear strain.

We model the measurement noise for each gage using an independent Gaussian distribution with zero mean and a standard deviation of $\sigma = 10\mu$ strain. This estimate of the standard deviation is obtained from published data that high-accuracy strain gages often have a 2–5% accuracy range when properly calibrated [34]. More sophisticated models of measurement noise, if available for a particular sensor implementation, can be incorporated in our approach.

Let the strain gage rosette readings at the kth sensor location be represented as the vector $\boldsymbol{\epsilon}^k = [\,\epsilon_1^k \quad \epsilon_2^k \quad \epsilon_3^k\,]^T$, where ϵ_1^k is the gage strain along axis 1, ϵ_2^k is the gage strain along axis 2, and ϵ_3^k is the gage strain along the 45 deg axis. Assuming our material coordinate 1 and 2 axes align perfectly with the rosette 1 and 2 axes, respectively (i.e., ignoring possible angular misalignment of the gages), $\boldsymbol{\epsilon}^k$ is related to the three-element plane strain vector at location k, $\varepsilon^k = [\,\varepsilon_{11}^k \quad 2\varepsilon_{12}^k \quad \varepsilon_{22}^k\,]^T$, by

$$\varepsilon^k = H\epsilon^k \tag{13}$$

¶The bisection process is significantly less efficient than the adaptive boundary construction from Sec. IV.B when considering a range of airspeeds, and is only reasonable here given that our test case is at one fixed airspeed.

where

$$H = \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 2 \\ 0 & 1 & 0 \end{bmatrix} \tag{14}$$

Thus, our measurement at the kth sensor location is defined as

$$\hat{\boldsymbol{s}}^k = \varepsilon^k + \mathsf{e}^k \tag{15}$$

where $\mathsf{e}^k \sim \mathcal{N}(0, \sigma^2 HH^T)$ is noise in the measured plane strain values at the kth sensor location due to strain gage measurement error. The full synthetic measurement (for all four sensor locations) is obtained as $\hat{\boldsymbol{s}} = [\hat{\boldsymbol{s}}^1; \ldots; \hat{\boldsymbol{s}}^4]$. Because each sensor is assumed to have independent noise, we can assemble the full measurement noise model $p(\hat{s}|D_j)$ for damage case D_j as the product

$$p(\hat{s}|D_j) = \prod_{k=1}^{4} p(\hat{s}^k|D_j) \tag{16}$$

Equation (16) is the measurement noise model for a single sensor sample. A practical application would yield multiple sensor measurements (for example, measurements accumulated over time). We consider the general case in which N_d data samples are available, $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{N_d}$. They are assumed to be independent and identically distributed. In this case, the modified measurement noise model is

$$p(\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{N_d}|D_j) = \prod_{\ell=1}^{N_d} p(\hat{s}_\ell|D_j) \tag{17}$$

Note that the accumulation of multiple samples will in general decrease the effects of sensor noise. In a practical situation, we could accumulate samples over a period of time, for instance, if $N_d = 10$, a sampling rate of 10 Hz would permit a capability estimate every 1 s. We note that a parameter such as the sensor sampling rate is highly system dependent, and so our exploration in the following results will compare in a relative sense the impact of the sample accumulation N_d.

B. Capability Estimator Performance

This subsection analyzes how the capability estimator behaves over the damage cases of interest. In particular, we assess whether the estimator can accurately reproduce the true capability as given by the value of $n_{\max}^{\text{truth}}$ for each damage case.

Table 1 Levels for each damage parameter used to construct 150 damage cases (two full-factorial explorations were performed)

Damage parameter	Trial 1 levels	Trial 2 levels
l_s	0.15	0.15
w_s	0.15	0.15
l_c	0.35	0.3, 0.45, 0.6
w_c	0.1, 0.2, 0.3, 0.4, 0.5	0.1, 0.2, 0.3, 0.4, 0.5
d_t	0.5, 0.6, 0.7, 0.8, 0.9	0.5, 0.6, 0.7, 0.8, 0.9
d_f	0.95, 0.96, 0.97	0.98

Table 2 Representative damage cases used for analyzing the online capability estimator behavior

Label	l_s	w_s	l_c	w_c	d_t	d_f	$n_{\max}^{\text{truth}}$	Included in online library?
$\boldsymbol{C}_0$	0	0	0	0	0	0	2.94	Yes
$\boldsymbol{C}_1$	0.15	0.05	0.6	0.1	0.7	0.98	2.59	Yes
$\boldsymbol{C}_2$	0.15	0.05	0.35	0.5	0.7	0.97	2.02	Yes
$\boldsymbol{C}_3$	0.15	0.05	0.35	0.1	0.5	0.95	2.53	No
$\boldsymbol{C}_4$	0.15	0.05	0.35	0.2	0.9	0.95	1.80	No

We test the estimator performance in the online phase for the cases in which the vehicle is in one of the five representative damage cases, $D \in \{\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{C}_4\}$. Synthetic data for each damage case are generated using the process described in Sec. V.A.3. We then use our approach to compute and plot the capability estimate $q(\boldsymbol{x}, \hat{\boldsymbol{s}})$ for $x \in \{(V, n) : V = 210 \text{ ft/s}, n \in [1, 3.5]\}$ The analysis is carried out for ten realizations of synthetic data for each damage case.

Figure 12 shows the capability estimator output for one sample run, which considers one realization of the $\boldsymbol{C}_2$ damage case. The dotted line shows the damage case's true maximum load factor $n_{\max}^{\text{truth}}$. The solid line is the $q(\boldsymbol{x}, \hat{\boldsymbol{s}})$ estimator output evaluated at an airspeed of $V = 210$ ft/s, reporting the probability that the label for the query state (V, n) is $+1$ (i.e., that the label for $\boldsymbol{x}$ is safe). Intuitively, a well-performing q will be close to 1 for low values of n and drop to zero as n crosses the maximum load factor for the damage case. In the following results, we will present similar plots for other damage cases. We refer to this curve as the PSVM output or the estimator output.

The number of records in the damage library can affect the quality of the capability estimate. We explore this behavior by down-sampling the number of library records and storing a sparser set for use in the online phase, and we hypothesize that more downsampling would degrade the capability estimate. We recall here that the original set of library records was generated using two full-factorial explorations of the damage parameters, and we are downsampling from this original set. Thus, the original sampling of the damage cases and the downsampled sets are not related to the adaptive sampling approach used to form the PSVM for each damage case. Figure 13 shows an example of the downsampling process. We define the downsampling ratio DSR to be the ratio of the number of original library records to the number of downsampled library records. For a given value of DSR, we conduct the following steps:

1) Order the full set of offline library records from least severe to most severe according to their prediction of the maximum safe load factor $n_{\max}$ at the fixed flight speed $V = 210$ ft/s for a PSVM probability value of 0.95.

2) Retain one record from each subset of size DSR records for storage in the online library.

3) If necessary, add the three example damage cases $\boldsymbol{C}_0, \boldsymbol{C}_1$, and $\boldsymbol{C}_2$ to the downsampled set.

We analyze the effect of varying the downsampling ratio DSR over the values {1, 10, 20, 40} (where $DSR = 1$ is the original, fully populated library). For each of the five damage test cases in Table 2, we accumulate $N_d = 10^3$ synthetic strain gage samples; the samples are then used with the downsampled library to produce a capability estimate (i.e., a PSVM output). We repeat the accumulation of samples and the ensuing analysis ten times. Figure 14 shows the resulting estimator outputs.

Based on these results, we note that the estimator had the most difficulty obtaining an accurate capability estimate in the $\boldsymbol{C}_0$ test case at the highest downsampling ratio. This is most likely because $\boldsymbol{C}_0$ is the undamaged test case, which has the highest maximum load factor and thus is on what could be considered the boundary of the damage case library; the estimator tends to classify into states that are either the undamaged state or worse, and thus it shows bias in the direction of load factors that are less than the undamaged one. This bias is particularly large when the damage library is sparse (high DSR).

In addition, the $\boldsymbol{C}_4$ damage case shows bimodal behavior, as the estimator tends to vary between two maximum load factors. These correspond to two damage cases in the library that are likely given the sensor readings but are not the true damage case; the true damage case $\boldsymbol{C}_4$ is not contained in the library, and the estimator does its best to interpolate using the available records.

Lastly, the estimator shows the best performance (both in terms of accurate prediction of the true maximum load factor as well as a consistent prediction) when the downsampling ratio is 1; that is, when the entire damage case library is used to classify the sensor readings.

C. Comparison to Static Capability Estimate

This subsection compares the performance of our dynamic data-driven capability estimator to a baseline case that uses a traditional static estimate of capability. The results in this subsection again consider the case in which the vehicle is operating at $V = 210$ ft/s. Our goal is to determine the maximum load factor at which an agent can operate the vehicle in a safe manner.

We benchmark against a case in which the agent uses a static capability estimate based off the known maximum load factor from design n_0. Note that n_0 is equal to $n_{\max}^{\text{truth}}$ evaluated for case $\boldsymbol{C}_0$ (i.e., the undamaged case). The agent then chooses to operate the vehicle at a maximum load factor $n_{\text{op}}^{\text{static}} \in [1, n_0)$. A value of $n_{\text{op}}^{\text{static}}$ near 1 indicates conservative behavior, while a value of $n_{\text{op}}^{\text{static}}$ near n_0 indicates aggressive behavior that operates the vehicle close to its undamaged limit. In our problem setup, choosing $n_{\text{op}}^{\text{static}} \geq n_0$ leads to exceeding the flight envelope.

In comparison, an agent using our dynamic capability estimate operates the vehicle at a maximum load factor that changes depending on the current sensor data. In this case, we denote the maximum load factor at which the agent chooses to operate the vehicle as n_{op}. To choose a value for n_{op}, the agent picks the largest load factor that has an acceptable probability of belonging to the vehicle capability set; we denote this acceptable probability as p_{op}. A value of p_{op} near 1 indicates conservative behavior (i.e., high confidence that the selected load factor will result in safe operation), whereas a value of

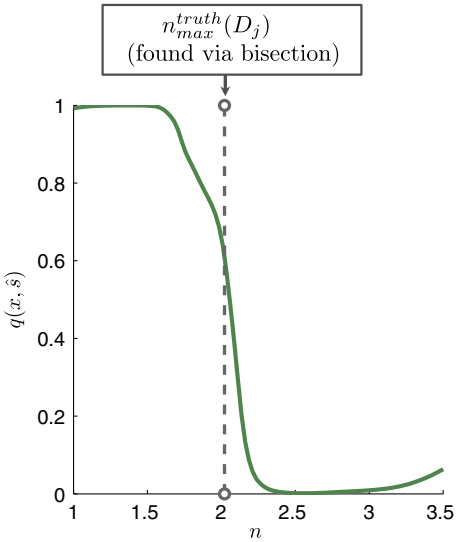


Fig. 12 The estimator $q(\boldsymbol{x},\hat{\boldsymbol{s}})$ evaluated at $V = 210$ ft/s for one realization of synthetic data for damage case $\boldsymbol{C}_2$. The estimator reports the probability that the query state is labeled as safe.

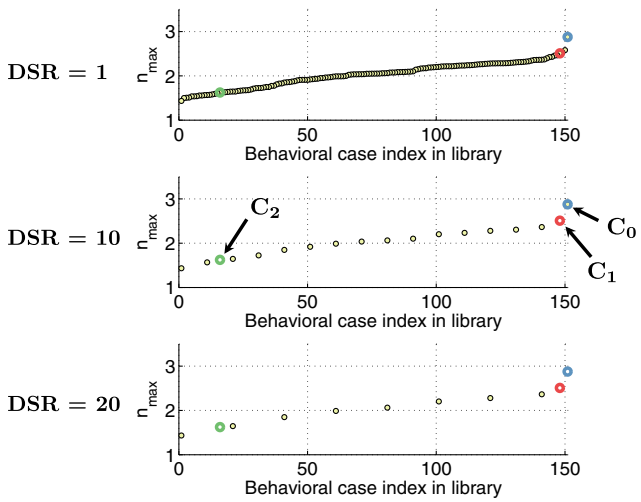


Fig. 13 The down-sampling ratio DSR controls the rate at which damage cases are retained from the complete library. The damage cases $\boldsymbol{C}_0$, $\boldsymbol{C}_1$, and $\boldsymbol{C}_2$ are always retained.

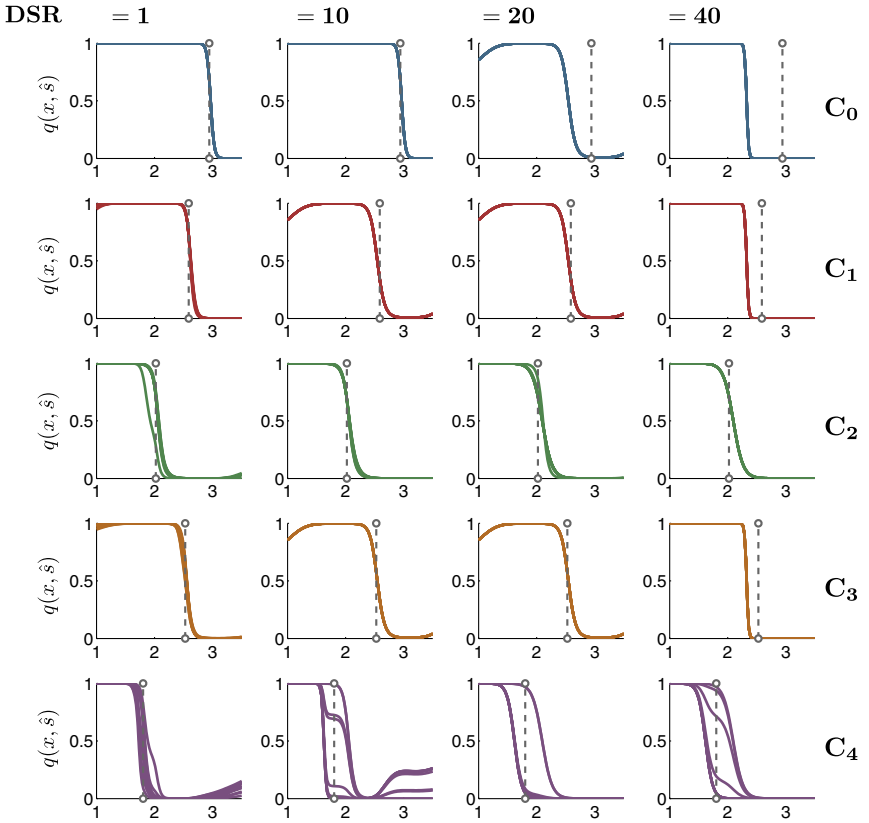


Fig. 14 Output from the capability estimator q over the load factor range $n \in [1,3.5]$ and fixed airspeed $V = 210$ ft/s, for $DSR \in \{1,10,20,40\}$ and five representative damage cases C_0, C_1, C_2, C_3, and C_4. Each experiment is repeated for ten independent trials.

p_{op} near zero indicates aggressive behavior (i.e., low confidence that the selected load factor will result in safe operation).

Figure 15 presents an example of how an agent would use the capability estimator output $q(\boldsymbol{x}, \hat{\boldsymbol{s}})$ to choose a value of n_{op}. In the case shown, setting $p_{\text{op}} = 0.9$ causes the agent to operate the vehicle at a maximum load factor of $n_{\text{op}} \approx 1.75$. For this case, the true maximum load factor $n_{\text{max}}^{\text{truth}} = 2.0$; thus, this choice does in fact result in safe operation. We compare the behavior of the static estimate with the dynamic estimate by varying p_{op} and $n_{\text{op}}^{\text{static}}$. We perform the following procedure:

1) Choose values for $n_{\text{op}}^{\text{static}}$ and p_{op}.

2) Set the vehicle library downsampling ratio DSR and the sample accumulation N_d to nominal values of 10 and 100, respectively (see Sec. V.B). Downsample the full vehicle library according to DSR and store it for continuing use.

3) For each example damage case D_j, $j = 1, 2, \ldots, R$ from the original full set of damage cases (before downsampling), accumulate N_d observable vector samples to form the synthetic sensor measurement $\hat{\boldsymbol{s}}$. Use the estimator output $q(\boldsymbol{x}, \hat{\boldsymbol{s}})$ and p_{op} to compute n_{op} (as shown in Fig. 15).

4) Repeat steps 2–3 for ten trials, and record the resulting values of $n_{\text{op}}^{\text{static}}$ and n_{op}, as well as the true maximum load factor for the current damage case $n_{\text{max}}^{\text{truth}}(D_j)$.

5) Plot $n_{\text{max}}^{\text{truth}}$ vs $n_{\text{op}}^{\text{static}}$ and n_{op}, overlaying all R damage case samples on the same plot.

This procedure is repeated over the values $n_{\text{op}}^{\text{static}} \in \{1, 1.5, 2, 2.5, 3\}$ for the static estimation case and over the values $p_{\text{op}} \in \{0.01, 0.26, 0.50, 0.74, 0.99\}$ for the dynamic case.

Figure 16 plots the resulting maximum load factor chosen by the agent vs the true maximum load factor. Each subplot corresponds to a different choice by the agent; the dynamic cases are labeled by the choice of p_{op}, whereas the static cases correspond to the agent's choice of $n_{\text{op}}^{\text{static}}$. Within each subplot, a sample point corresponds to an operation of the vehicle given a vehicle damage case and a realization of the observable vector sample. The black lines with slope 1 indicate the boundary between successful and unsuccessful operations; the samples below the line are successful because $n_{\text{op}} < n_{\text{max}}^{\text{truth}}$, and the samples above the line are unsuccessful because $n_{\text{op}} \geq n_{\text{max}}^{\text{truth}}$.

In the static case, the agent chooses one value of $n_{\text{op}}^{\text{static}}$ and is uninformed of any potential aircraft capability changes due to structural weakness. Hence, many trials fail, especially when the agent is more aggressive and/or significant levels of structural weakness are present. The dynamic cases use our capability estimation strategy; depending on the value of p_{op}, the agent is either less or more

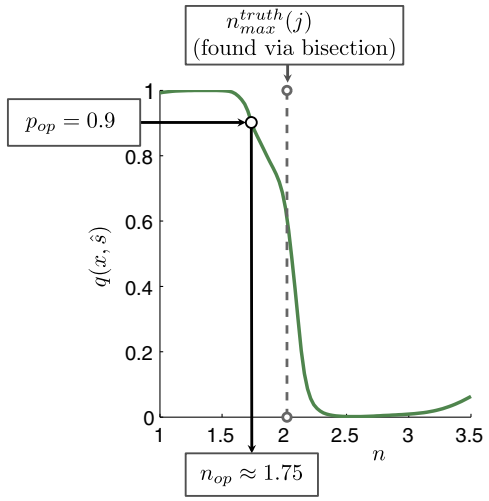


Fig. 15 The agent specifies p_{op} according to their degree of conservativeness when using the capability estimator output. p_{op} then determines the maximum operating load factor given the dynamic data-driven estimator output.

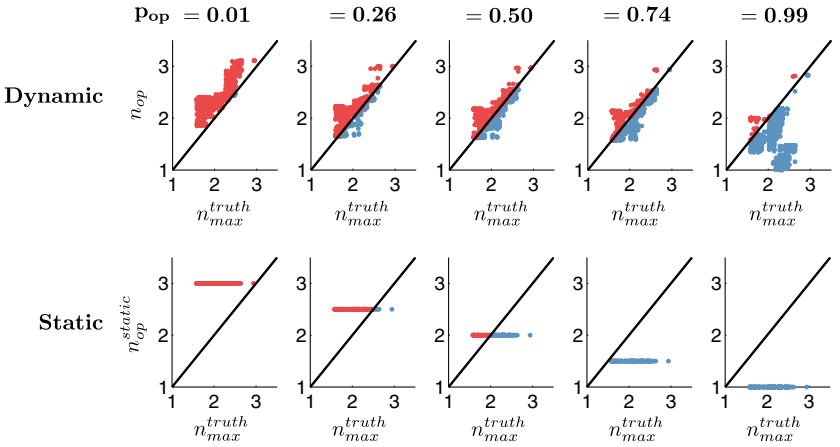


Fig. 16 Maximum load factor using static and dynamic estimation strategies for ten sensor realizations in each of the library damage cases. For samples above the line the decision led to failure.

conservative. The value $p_{\text{op}} = 0.01$ is highly unlikely as a choice in a practical situation. The results are shown for illustrative purposes; it can be seen that this aggressive behavior leads to almost all operations failing. As the value of p_{op} increases, the percentage of successful operations increases. With a sufficiently conservative value of p_{op} (e.g., $p_{\text{op}} = 0.99$), our dynamic capability estimate leads to most operations being executed successfully, at the cost of operating at a lower average value of n_{op}.

D. Reliability of Dynamic Capability Estimation

In this subsection, we analyze how the decision strategy, informed by the capability estimate, trades off between survivability and full utilization of the vehicle capability. We quantify survivability by evaluating probability of maneuver success for each value of p_{op} over all possible damage cases. We quantify utilization using the average ratio between the vehicle's operational load factor and the vehicle's true maximum load factor.

The probability of maneuver success is defined as the probability that the agent chooses a value of n_{op} that is less than the maximum vehicle load factor $n_{\text{max}}^{\text{truth}}$. Note that here our estimate of the probability of maneuver success is based on whether we exceed the flight envelope; in practice, the flight envelope is generated with a substantial safety factor, and thus any onboard decision making (e.g., mission replanning) might also take that into account. We denote the event that the agent succeeds in the maneuver as MS, where

$$\text{MS} = \{n_{\text{op}}(\mathsf{D}, \hat{\mathbf{s}}) < n_{\text{max}}^{\text{truth}}(\mathsf{D})\} \tag{18}$$

The probability of maneuver success $p(\text{MS})$ is computed as

$$p(\text{MS}) = \sum_{j=1}^{R} p(\text{MS}|D_j)p(D_j) \tag{19}$$

The quantity $p(\text{MS}|D_j)$ is the probability the agent succeeds given the vehicle is in the damage state $\mathsf{D} = D_j$; it is approximated by the fraction of ten trials for damage case D_j that are successful. For this analysis, we assume a uniform prior over all the damage cases so $p(D_j) = 1/R$ for all D_j.

The utilization is defined as the average ratio between the vehicle's operational load factor and the vehicle's true maximum load factor. This is the expected value of $n_{\text{op}}(\mathsf{D}, \hat{\mathbf{s}})/n_{\text{max}}^{\text{truth}}(\mathsf{D})$ conditioned on the event MS. The conditioning is because the vehicle capability is only used when it does not exceed the flight envelope; only the cases in which the agent chooses a safe maximum load factor contribute positively to the utilization of the vehicle capability. We denote this metric as $\bar{n}_{\text{util}}$ and compute it as

$$\begin{aligned}\bar{n}_{\text{util}} = E\left[\frac{n_{\text{op}}(\mathsf{D}, \hat{\mathbf{s}})}{n_{\text{max}}^{\text{truth}}(\mathsf{D})}|\text{MS}\right] &= \sum_{j=1}^{R} E\left[\frac{n_{\text{op}}(D_j, \hat{\mathbf{s}})}{n_{\text{max}}^{\text{truth}}(D_j)}|\text{MS}, D_j\right]p(D_j) \\ &= \sum_{j=1}^{R} \frac{E[n_{\text{op}}(D_j, \hat{\mathbf{s}})|\text{MS}, D_j]}{n_{\text{max}}^{\text{truth}}(D_j)}p(D_j)\end{aligned} \tag{20}$$

The quantity $E[n_{\text{op}}(D_j, \hat{\mathbf{s}})|\text{MS}, D_j]$ is the mean of the agent's chosen n_{op} given that the vehicle is in the damage case D_j and that n_{op} is less than $n_{\text{max}}^{\text{truth}}$. We approximate this value using the sample mean of all successful trials out of the ten total that were conducted for $\mathsf{D} = D_j$ [i.e., the same set of samples from the previous $p(\text{MS})$ computation].

Figure 17 plots the probability of maneuver success $p(\text{MS})$ vs the utilization $\bar{n}_{\text{util}}$ for the two decision strategies. A third curve is plotted that shows the performance of a hypothetical estimator that knows the damage case with absolute certainty; that is, its only error is due to the approximation of the corresponding capability set using the PSVMs.

The data shown in this figure were generated using 500 equally spaced values of p_{op} from 0.01 to 0.99 and 500 equally spaced values of $n_{\text{op}}^{\text{static}}$ from 1 to 3. The ideal decision strategy would have both perfect usage of available capability (i.e., $\bar{n}_{\text{util}} = 1$) and certain maneuver success [i.e., $p(\text{MS}) = 1$]; this is marked as the Utopia point in the upper right corner of Fig. 17. A sample on the plot is considered to be nondominated if no other sample has both a higher $p(\text{MS})$ and higher $\bar{n}_{\text{util}}$ value (or higher value of one and equal value of the other); the nondominated combinations of $[n_{\text{util}}, p(\text{MS})]$ for each estimator are connected by a dashed line of the corresponding color.

Figure 17 shows that the static capability case has $p(\text{MS}) = 1$ at values of $\bar{n}_{\text{util}} < 0.75$. This is because if the agent sets a sufficiently low static load factor the realized load is less than $n_{\text{max}}^{\text{truth}}(D_j)$ for all $j = 1, \ldots, R$. Thus, within the scope of our analysis, the vehicle never exceeds the flight envelope, although the loads are limited to conservative values and utilization is low. The figure also shows that the static capability case has a long trail of samples near $p(\text{MS}) = 0$ at high values of $\bar{n}_{\text{util}}$. This is because at values of $n_{\text{op}}^{\text{static}}$ close to 2.9 the vehicle almost certainly exceeds the flight envelope unless it is in the pristine case, which has a small probability (<0.01) of occurring. We see that the dynamic estimator results in an even spread of points across the nondominated front, with a sharp "knee" at $\bar{n}_{\text{util}} \approx 0.95$ where the probability of maneuver success drops rapidly.

We can use the nondominated fronts, Fig. 17, as a measure of performance of each capability estimate when used for decision making in the flight scenario. For instance, if the agent wants to use 95% of the maximum vehicle load factor on average, there would be an 80% chance of maneuver success using the dynamic estimate of the load factor as opposed to a 40% chance of success when operating at a static load factor. On the other hand, if the agent can accept operating at less than 80% of the maximum capability on average, then both estimators show similar performance. We note this is most

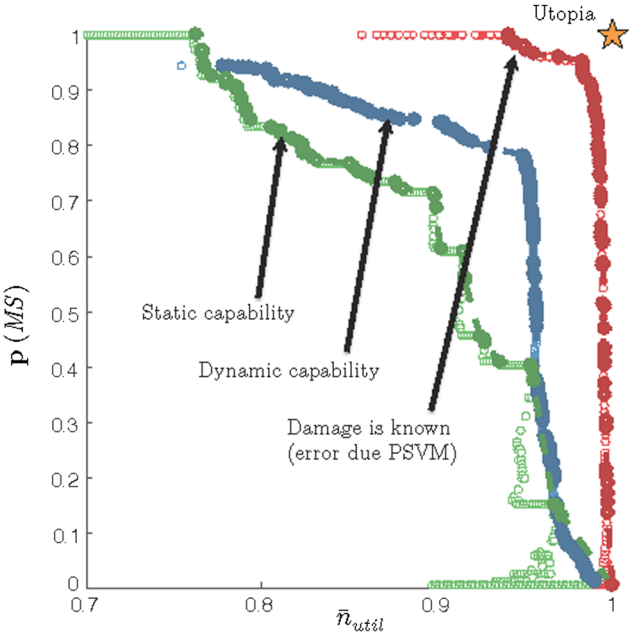


Fig. 17 Probability of maneuver success versus average fraction of vehicle capability used for an onboard decision process agent using static and dynamic capability estimation strategies.

likely because the damage cases in the library cause a limited reduction in the vehicle capability, and simulating more severe damage cases would continue to emphasize the performance gain from using the dynamic capability estimate. Lastly, Fig. 17 shows that if the damage were known perfectly, the error introduced by the PSVM approximations would be relatively small for this example. The difference between the dynamic capability curve and the known damage curve is an indication of the value (in terms of vehicle capability utilization and maneuver success probability) of increased quality and quantity of sensors. Comparisons of this kind could be used to support design decisions regarding the cost vs value tradeoffs of including more onboard sensors.

VI. Conclusions

This paper explored the concept of dynamic capability estimation from a data-driven perspective, in which models and experimentation can both be sources of information and where the vehicle behavior is analyzed cognizant of model uncertainty. A key to the approach is an offline/online decomposition of tasks so that the predictive power of high-fidelity physics-based models can be leveraged while achieving rapid estimations in the face of dynamic data. Results demonstrated the benefit of incorporating vehicle sensor information into a dynamically updated estimate of the structural capability. This dynamically updated estimate is computed here using the observable vector sample to classify current vehicle behavior followed by the probabilistic classification of vehicle capability. Future research could explore the possibility of constructing high-dimensional support vector machines on the state and observable vector space to create a direct map between sensor readings and capability sets. Doing so would remove the need to associate sensor readings to library records. This might lead to less insight with respect to the physical model but might result in computational gains. This kind of direct mapping has been explored using proper orthogonal decomposition representations and self-organizing maps in Ref. [16]. Lastly, note that, although the aircraft application presented relies on computational models to generate training data, the methodology is general and also permits the incorporation of experimental data.

Acknowledgments

This work was supported by U.S. Air Force Office of Scientific Research grant FA9550-11-1-0339 under the Dynamic Data-Driven Application Systems Program, Program Manager Frederica Darema. The authors thank Carlos Cesnik for use of the University of Michigan Variational Asymptotic Beam Section Analysis, revision 1.31. The authors also thank David Kordonowy and Jeff Chambers at Aurora Flight Sciences for the use of their unmanned aerial vehicle design software and for providing domain-specific knowledge in aircraft structural health monitoring.

References

[1] Brintrup, A. M., Ranasinghe, D. C., Kwan, S., Parlikad, A., Owens, K., and Company, T. B., "Roadmap to Self-Serving Assets in Civil Aerospace," *Proceedings of the 1st CIRP Industrial Product-Service Systems (IPS2) Conference*, Cranfield Univ., Cranfield, Bedford, U.K., April 2009, p. 323, http://dspace.lib.cranfield.ac.uk/handle/1826/3873 [retrieved 17 May 2015].

[2] Willis, S., "OLM: A Hands-On Approach," *ICAF 2009, Bridging the Gap Between Theory and Operational Practice*, Springer–Verlag, Berlin, 2009, pp. 1199–1214.

[3] Staszewski, W., Tomlinson, G., and Boller, C., *Health Monitoring of Aerospace Structures Smart Sensor Technologies and Signal Processing*, 1st ed., Wiley, Hoboken, NJ, 2004.

[4] Benedettini, O., Baines, T. S., Lightfoot, H. W., and Greenough, R. M., "State-of-the-Art in Integrated Vehicle Health Management," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 223, No. 2, 2009, pp. 157–170. doi:10.1243/09544100JAERO446

[5] Ortiz, E. M., Clark, G. J., Babbar, A., Vian, J. L., Syrmos, V. L., and Arita, M. M., "Multi Source Data Integration for Aircraft Health Management," *2008 IEEE Aerospace Conference*, IEEE Publ., Piscataway, NJ, 2007, pp. 1–12. doi:10.1109/AERO.2008.4526625

[6] Gorinevsky, D., Mah, R., Srivastava, A., Smotrich, A., Keller, K., and Felke, T., "Open Architecture for Integrated Vehicle Health Management," *AIAA Infotech@Aerospace 2010*, AIAA Paper 2010-3434, 2010.

[7] Fox, J., and Glass, B., "Impact of Integrated Vehicle Health Management (IVHM) Technologies on Ground Operations for Reusable Launch Vehicles (RLVs) and Spacecraft," *Proceedings of 2000 Institute of Electrical and Electronics Engineers Aerospace Conference*, Vol. 2, IEEE Publ., Piscataway, NJ, 2000, pp. 179–186.

[8] Kirk, B., Schagaev, P. I., Wittig, T., Kintis, A., Kaegi, T., Friedrich, F., Ag, E. T., Spirit, S. A., Avenue, S., and Faliro, P., "Active Safety for Aviation," *6th INO Workshop*, EUROCONTROL Experimental Centre, Brétigny-sur-Orge, France, 2007.

[9] Mehr, A. F., Tumer, I., and Barszcz, E., "Optimal Design of Integrated Systems Health Management (ISHM) for Improving the Safety of NASA's Exploration Missions: A Multidisciplinary Design Approach," *6th World Congresses on Structural and Multidisciplinary Optimization*, International Soc. for Structural and Multidisciplinary Optimization (ISSMO), Dept. of Mechanical Engineering, IST–Instituto Superior Tecnico, Lisboa, Portugal, May–June 2005, http://www.issmo.net [retrieved 17 May 2015].

[10] Sohn, H., and Farrar, C., "Damage Diagnosis Using Time Series Analysis of Vibration Signals," *Smart Materials and Structures*, Vol. 10, No. 3, 2001, pp. 446–451. doi:10.1088/0964-1726/10/3/304

[11] Farrar, C., and Lieven, N., "Damage Prognosis: The Future of Structural Health Monitoring," *Philosophical Transactions of the Royal Society of London, Series A: Mathematical and Physical Sciences*, Vol. 365, No. 1851, 2007, pp. 623–632. doi:10.1098/rsta.2006.1927

[12] Farrar, C., Doebling, S., and Nix, D., "Vibration-Based Structural Damage Identification," *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, Vol. 359, No. 1778, 2001, pp. 131–149. doi:10.1098/rsta.2000.0717

[13] Prudencio, E., Bauman, P., Williams, S., Faghihi, D., Ravi-Chandar, K., and Oden, J., "A Dynamic Data Driven Application System for Real-Time Monitoring of Stochastic Damage," *Procedia Computer Science*, Vol. 18, 2013, pp. 2056–2065. doi:10.1016/j.procs.2013.05.375

[14] Raymer, D., *Aircraft Design: A Conceptual Approach*, 3rd ed., AIAA, Reston, VA, 1996.

[15] Kordonowy, D., and Toupet, O., "Composite Airframe Condition-Aware Maneuverability and Survivability for Unmanned Aerial Vehicles," *Infotech@Aerospace 2011*, AIAA Paper 2011-1496, 2011.

[16] Mainini, L., and Willcox, K., "A Surrogate Modeling Approach to Support Real-Time Structural Assessment and Decision-Making," *AIAA Journal*, Vo. 53, No. 6, 2015, pp. 1612–1626; also *10th AIAA Multidisciplinary Design Optimization Conference*, AIAA Paper 2014-1488, Jan. 2014.

[17] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification*, 2nd ed., Wiley, New York, 2000.

[18] Greitzer, E. M., Bonnefoy, P. A., Blanco, E. D. l. R., Dorbian, C. S., Drela, M., Hall, D. K., Hansman, R. J., Hileman, J. I., Liebeck, R. H., Lovegren, J., Mody, P., Pertuze, J. A., Sato, S., Spakovszky, Z. S., Tan, C. S., Hollman, J. S., and Kordonowy, D., "$N + 3$ Aircraft Concept Designs and Trade Studies, Final Report Volume 1," NASA CR-2010-216794, 2010.

[19] Drela, M., "Integrated Simulation Model for Preliminary Aerodynamic, Structural, and Control-Law Design of Aircraft," *Proceedings of the 40th AIAA SDM Conference*, AIAA Paper 1999-1394, 1999.

[20] Cesnik, C. E. S., and Hodges, D. H., "VABS: A New Concept for Composite Rotor Blade Cross-Sectional Modeling," *Journal of the American Helicopter Society*, Vol. 42, No. 1, 1997, pp. 27–38. doi:10.4050/JAHS.42.27

[21] Palacios, R., and Cesnik, C. E., "Cross-Sectional Analysis of Non-homogeneous Anisotropic Active Slender Structures," *AIAA Journal*, Vol. 43, No. 12, 2005, pp. 2624–2638. doi:10.2514/1.12451

[22] Boczko, E. M., Xie, M., Wu, D., and Young, T., "Comparison of Binary Classification Based on Signed Distance Functions with Support Vector Machines," *Bioinformatics, 2009 Ohio Collaborative Conference*, IEEE Publ., Piscataway, NJ, 2009, pp. 139–143.

[23] Cortes, C., and Vapnik, V., "Support-Vector Networks," *Machine Learning*, Vol. 20, No. 3, 1995, pp. 273–297. doi:10.1023/A:1022627411411

[24] Aizerman, A., Braverman, E. M., and Rozoner, L. I., "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Automation and Remote Control*, Vol. 25, 1964, pp. 821–837.

[25] Platt, J. C., "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, 1999, pp. 61–74.

[26] Lin, H.-T., Lin, C.-J., and Weng, R. C., "A Note on Platt's Probabilistic Outputs for Support Vector Machines," *Machine Learning*, Vol. 68, No. 3, 2007, pp. 267–276. doi:10.1007/s10994-007-5018-6

[27] Basudhar, A., "Computational Optimal Design and Uncertainty Quantification of Complex Systems Using Explicit Decision Boundaries," Ph.D. Thesis, Univ. of Arizona, Tucson, AZ, 2011.

[28] McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239–245. doi:10.2307/1268522

[29] Du, Q., Faber, V., and Gunzburger, M., "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, Vol. 41, No. 4, 1999, pp. 637–676. doi:10.1137/S0036144599352836

[30] Lloyd, S., "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28, No. 2, 1982, pp. 129–137. doi:10.1109/TIT.1982.1056489

[31] Basudhar, A., and Missoum, S., "An Improved Adaptive Sampling Scheme for the Construction of Explicit Boundaries," *Structural and Multidisciplinary Optimization*, Vol. 42, No. 4, 2010, pp. 517–529. doi:10.1007/s00158-010-0511-0

[32] Basudhar, A., and Missoum, S., "Adaptive Explicit Decision Functions for Probabilistic Design and Optimization Using Support Vector Machines," *Computers and Structures*, Vol. 86, Nos. 19–20, Oct. 2008, pp. 1904–1917. doi:10.1016/j.compstruc.2008.02.008

[33] "Strain Gage Rosettes: Selection, Application, and Data Reduction," Micro-Measurements, Vishay Precision Group, TN-515, Malvern, PA, Aug. 2014, http://www.vishaypg.com/docs/11065/tn-515.pdf [retrieved 17 May 2015].

[34] "Strain Gage Selection: Criteria, Procedures, and Recommendations," Vishay Precision Group, TN-505-4, Malvern, PA, Aug. 2014, http://www.vishaypg.com/docs/11055/tn505.pdf [retrieved 17 May 2015].

R. Haftka
Associate Editor